Logistics

• **Questions**
  - Have you confirmed your CS accounts?

• **Computer Access Forms**
  - NERSC forms please read, sign and return them (to Ricky) on Friday (1/14/2004).
  - SCL please follow the link on the class web page or go to the following URL:
    - http://www.scl.ameslab.gov/scl_user_info/policy.html
    - Print, read, sign, and return them (to Ricky) on Friday (1/14/2004).

• **Chapter 1 is the reading assignment for the week.**
  - Check the course web page for the assignment.
Logistics (NERSC Form)

- I have read the NERSC Policies and Procedures and understand my responsibilities in the use of NERSC resources.

Signature: <data>
Print Name: <data>
Citizenship: <data>
Organization: Iowa State University
Email Address: <data>
Work Phone Number: <data: daytime phone>
Principal Investigator: Ricky A. Kendall
Date: <data>
Logistics (SCL Form)

• I understand the computer use policies above and agree to follow them when using Scalable Computing Laboratory Resources.

• Name:__________________________________

• Telephone:______________________________

• Email Address:__________________________

• Sponsor's Name: Ricky A. Kendall

• Organization: ComS/CprE 425 ISU

• Signature:_______________________________

• Date:___________________________________

• Access Needed: 4pack, redwing, osage
Logistics Homework

• Everyone started working on the homework?
  ➢ Design documents should be done first!
  ➢ Coding should follow the design.
  ➢ If the code differs from the design is there a problem?

• Website messages list.
  ➢ A link off the main page.
  ➢ Will be updated as questions are asked or problems noted.
  ➢ Updated with queries from students to me or Bin Tong (your TA).
Logistics Homework [2]

- **General Homework Instructions**
  - You need to turn in your design document and your results and analysis document, along with the software, job execution output(s), makefiles etc.
  - Each function you write should be in a separate file and properly identified with the names of the student(s) in comments near the top of the file!
    - This will make it easier to pick and choose on reuse later.
    - It will make it easier for me to grade as well.
  - Also the first thing that your program should do is to print out each student's name in the group.
  - **First Homework is done individually**
  - Teams will be formed after I review the first homework!
Logistics Homework [3]

• What is wrong with this code?

```c
#include <stdio.h>
#include "protos.h"
#include "function1.c"
#include "function2.c"
#include "function3.c"
int main(int argc, char *argv[])
{
    ...
    return 0;
}
```

You do not include source code files!!!
Logistics Homework [4]

Instructions for turning in your Homework Assignments

• You need to use `turnin` to electronically hand in your homework.
  ➢ The script only runs on a few of the departmental machines. (Will fix this)
• To effectively make this work:
  1. log into popeye
  2. mkdir turnin
  3. cd turnin
  4. scp files_from somewher .
  5. ~/cs425/public/bin/turnin cs425 turnin_keyword
     or /usr/local/bin/turnin (on the SCL systems)
• files_from somewher are the files you want to turn in for your homework
  assignment.
  ➢ Please use filenames without spaces and for my sanity all Microsoft Word
    documents should end in .doc (also make sure you use scp or binary ftp to
    move Word Documents around), all latex/tex documents in .tex and all text
    documents should end in .txt.
• turnin_keyword is the keyword designated in the assignment and in the
  table of assignments on the main homework page.
• Please **DO NOT** turn in object files or the executable you generate, I will
  regenerate them when grading the homework assignment.
The algorithm

• This semester we will consider only a single algorithm.
  ➢ Matrix multiplication!
• The techniques you apply to this algorithm are essentially generic so you may apply them to other algorithms.
  ➢ How well your other algorithms work is a function of how well you understand the techniques of the course.
    ▪ Designing algorithms or the theoretical side of computer science is covered in many of your other courses.
  ➢ This is an applied Computer Science course.
Homework

- Homework #1 and #2 are posted on the class Home Page.
  - The due date are posted as well.
- The first two homeworks are very important.
  - Allows students and instructor/TA to provide mutual feedback and set expectations
  - Will be the base modules needed for most of the rest of the course.
    - If you don’t get it done you will have to do it for future assignments anyway 😊
  - Each homework builds on the next in some way.
  - All homeworks are based on this first one!
Matrix Multiply

• **Why we are using this algorithm?**
  - At the core of many other linear algebra algorithms.
  - Target core algorithm for high performance applications.
  - It is the “hello world” kind of program for high performance computing
  - Simple enough that it is easy to understand.
  - Complex enough that it exposes the nuances of most programming models.
  - Usually will lead to high performance.
  - Can be done in many different ways
    - Simple loops
    - Vector operations
    - Blocked operations
What is a matrix?

- A collection of numbers arranged in rows and columns.
- An ordered set of information with some specific relationship.
  - For this class we will consider 1 and 2 dimensional matrices.
    - A one dimensional matrix is a vector.
- An m by n matrix has “m” rows and “n” columns
- The matrix has m•n elements.
- Each element can be identified by it’s row index and column index.
  - E.g., $A_{i,j}$ is the element of the matrix A in the $i^{th}$ row and the $j^{th}$ column.
How do You Store a Matrix?

- **Standard 2-dimensional matrix in C**
  - $A[i][j]$ represents element $A_{i,j}$ of matrix A.
  - Each Row in the matrix must be allocated.
    - Via `malloc()` or new
  - Each Row has “contiguous” memory but the Matrix A may or may not be contiguous!

- **Store entire matrix in a single array**
  - Contiguous storage for the matrix
  - Requires an index function to map the “i” and “j” values to the appropriate array index.
  - $A[\text{mapfunction}(i,j,...)]$ represents element $A_{i,j}$.
    - The `mapfunction(i,j,NC) = i*NC+j`
      - NC is the number of columns in the matrix
    - Should this be a real function or a C/C++ macro??
One dimension or two?

• Reasons for malloc and index into the array usage:
  - Better cache performance control by the developer?
  - Contiguous block of memory for the matrix, patch of matrix, anything with two indices.
    - Later on when we do MPI, GA
      ♦ MPI_routine (&buffer, ....);
  - Question: with the multiple C index strategy can you hand a single pointer entity to a routine and have all the data behind the routine visible??
    - You may have to “pack” buffers with data before a send and “unpack” buffers with data after a receive
      ♦ Parallel overhead
Requirements

• **Homework #1**
  - It is very important that the interface to all three matrix generation routines should be such that you can get (in a single call to the generation routine):
    - a single element,
    - a single row or column,
    - or a block of each matrix
  - Why is this important??
    - In future homework assignments each thread or parallel process will operate, store, or manipulate only on a portion of the entire matrix.
  - How can this be accomplished in a single interface?
    - One way is: `gen_A(…, ilo, ihi, jlo, jhi, …)`
    - This will work nicely with your blocked algorithm.
Generation Routines

• Should they also do the memory allocation?
  ☐ Yes if you want to be object oriented!
  ☐ No if you want the most flexibility.

• Uncoupled memory management from the functionality is sometimes useful.
  ☐ Consider allocating two patches of a matrix that need to be in contiguous storage.

  ▪ With memory allocation in the generation routine
    ♦ Returns a pointer
    ♦ 1) generate patch one and generate patch two
    ♦ 2) allocate the “2 patch segments” and copy patches, and free.

  ▪ Without memory allocation in the generation routine
    ♦ Returns an error condition or void
    ♦ 1) Allocate “2 patch segments”
    ♦ 2) Pass appropriate pointers to generation routine.
Generator with memory allocation

What we want!

Generate blue patch

Generate green patch

copy

copy

copy
Generator without memory allocation

What we want!

Blue Generator

Green Generator
How do you multiply two matrices?

• A row of A times a column of B gives you an element of C

\[ C_{i,j} = \sum_k A_{i,k} B_{k,j} \]
Matrix Multiply
(Simple three loops)
Matrix Multiply

- $A \times B = C$
  - Number of columns of $A$ must equal Rows of $B$
    - $\text{coldim}A = \text{rowdim}B$
  - $C$ has same number of rows as $A$
  - $C$ has same number of columns as $B$
    - $\text{rowdim}A = \text{rowdim}C \& \text{coldim}B = \text{coldim}C$

```c
for (i=0;i<\text{rowdim}C;i++)
    for(j=0;j<\text{coldim}C,j++)
        for (k=0;k<\text{coldim}A;k++)
            C[i][j] += A[i][k] \times B[k][j];
```

- Mini-Quiz for enlightenment.
  - Analyze this algorithm and figure out why this will not perform optimally as written!!
Matrix Multiply Homework

\[
\begin{align*}
\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} B \end{bmatrix} & = \begin{bmatrix} C \end{bmatrix} \\
\begin{bmatrix} A \end{bmatrix} \begin{bmatrix} B \end{bmatrix} & = \begin{bmatrix} C \end{bmatrix}
\end{align*}
\]
Matrix Multiply Homework

Compute the Product

\[ A \times B \rightarrow C \]

- Elements of A and B are numbers
  - The values could be randomly generated
  - The values could be pre-determined

Via Some Algorithm
- Ddot Loops
- Daxpy Loops
- Strassen’s Algorithm
- Parallel Daxpy
- Cannon’s Algorithm
- Fox’s or BMR Algorithm
- Any Algorithm

1/12/2005

ComS 425
Spring 2005
23 of 37: Lecture 2
Analytic Matrix used in Homework

e.g, predetermined

\[ A = ai + bj + c \]
\[ B = di + ej + f \]

\[ C = \sum_{k=1}^{k_{\text{limit}}} A_{i,k} B_{k,j} \]

\[ = \sum_{k=1}^{k_{\text{limit}}} (ai+bk+c)(dk+ej+f) \]
Matrix Multiply Homework

(an specific example)

\[ A = 2i + (3/2)j \]

\[ B = 3i - (2/3)j \]

\[ C = \sum_{k=1}^{k_{\text{limit}}} [k(6i + (9/2)k - j) - (4/3)ij] \]
Matrix Multiply Homework
(Two different resultant matrices)

This matrix is computed via a matrix multiplication algorithm.
There are many of them
You will code 3 different ones for homework #2.

This is determined from the definition of the analytical constants that define the matrices A and B and the mathematical definition of matrix multiplication.
Matrix Multiply Homework

(Check to see that your algorithm works)

\[ \text{Computed} \quad - \quad \text{Analytical} \quad = \quad \text{zero} \]

If your computational algorithm worked then the computed matrix should be identical to the resultant analytical matrix!

\[ \text{fatal error} \quad = \quad \text{non zero} \]
Matrix Multiply Thinking

\[
\begin{align*}
A & \quad B & = & & C \\
A & \quad B & = & & C
\end{align*}
\]
Matrix Multiply (loops)

- $A \times B = C$
  - $\text{coldim}A == \text{rowdim}B$
  - $\text{rowdim}A == \text{rowdim}C$
  - $\text{coldim}B == \text{coldim}C$

  for $(i=0; i<\text{rowdim}C; i++)$
    for $(j=0; j<\text{coldim}C, j++)$
      for $(k=0; k<\text{coldim}A; k++)$
        $C[i][j] += A[i][k] \times B[k][j]$;

The loops algorithm is sometimes called the ddot algorithm!
Matrix Multiply (Vector [ddot])
Matrix Multiply (daxpy)

- \( A \times B = C \)
  - \( \text{coldimA} == \text{rowdimB} \)
  - \( \text{rowdimA} == \text{rowdimC} \) & \( \text{coldimB} == \text{coldimC} \)

\[
\text{for (i=0; i<rowdimC; i++)} \\
\quad \text{for (k=0; k<\text{coldimA}; k++)} \\
\quad \quad \text{const} = A[i][k]; \\
\quad \text{for (j=0; j<\text{coldimC}; j++)} \\
\quad \quad C[i][j] += \text{const} \times B[k][j];
\]

This is a simple loop rearrangement!
Matrix Multiply (daxpy)

\[ A \times B = C \]
Matrix Multiply

• When does performance drop off for both of these algorithms?
  – When the data sizes increase the cache miss rate!

• What can be done to solve this problem?
  ❖ Using a blocked algorithm where all the data fits into cache!
Matrix Multiply (blocked)

\[
\begin{align*}
A & \quad B & = & \quad C \\
A & \quad B & = & \quad C
\end{align*}
\]
Matrix Multiply (Blocked)

• Has a more complex loop structure
  ➢ Turns 3 loops into essentially 6 loops
  ➢ This would be a good to understand.
    ▪ You have to do it for the Homework
    ▪ You might be requested to do this in the future (e.g., on an exam).
Blocking a loop

• Loop to block:
  for (j=0; j < N; j++)
   { x = f(j); }

• To block the loop you have to determine
  - The size of the block to use
  - The mechanism to implement the appropriate values in the code.
  - This makes one loop into two loops.
  - The size of the block must be small enough to minimize cache misses
  - The size of the block must be large enough to get good performance!
Blocking a single loop

• Loop to block:  (j goes 0,1,2,3,4,…, N)
  for (j=0; j < N; j++)
  { x = f(j); }

• This becomes (one of many ways):
  block_size = 4;  num_blocks = N/block_size+1;
  for (jb = 0; jb < num_blocks; jb++) {
    jlo = jb*block_size;
    jhi = MYMIN(jlo+block_size, N)
    for (j=jlo; j<jhi; j++) { x = f(j); }
  }

(j goes 0,1,2,3, 4,5,6,7, 8,9,10,11, … N)